GPGPU implementation of modal parameter tracking by particle based Kalman filter

Antoine Crinière, Meriem Zghal, Laurent Mevel, Jean Dumoulin

GPGPU implementation of modal parameter tracking by particle based Kalman filter

Antoine Crinière[1], **Meriem Zghal**[1], **Laurent Mevel**[1], **Jean Dumoulin**[12]

[1] Inria/IFSTTAR, I4S Team, Campus de Beaulieu, 35042 Rennes cedex, France
[2] LUNAM Université, IFSTTAR, COSYS, SII, F-44340 Bouguenais, France

Keywords: Vibration monitoring, Particular filtering, Kalman, GPU

Abstract

This paper presents a method based on the use of Bayesian modal parameter recursive estimation based on a particular Kalman filter algorithm with decoupled distributions for mass and stiffness. Particular Kalman filtering is a combination of two widely used Bayesian estimation methods working together: the particle filter (also called sequential Monte Carlo samplings) and the Kalman filter. Usual system identification techniques for civil and mechanical structures assume the availability of large set of data derived from a stationary quasi steady structure. On the opposite, several scenarios involve time varying structures. For example, due to interaction with aerodynamics in aeronautics, some critical parameter may have to be monitored, for instability monitoring (leading possibly to flutter) of in flight data due to fuel consumption and speed change. This relates to the monitoring of time varying structural parameters such as frequencies and damping ratios. The main idea of a particular Kalman filter is to consider stochastic particles evolving in the parameter space. For each particle, a corresponding linear state is recursively estimated by applying a Kalman filter to the mechanical system, whose modal parameters are driven by the evolution of this time-varying particle. The weight of each particle is computed from the likelihood of the parameter sample it represents and its corresponding state. This result in a bank of adaptive coupled Kalman filters combined with their particle filter. However, the system parametrization is relatively large. In order to provide fast and convincing results for large time varying structure, such as an airplane, the execution time of the method has to be improved. In particular, the particle evolutions can be run in parallel, Within the Cloud2sm project, A Quadro k6000 card of 3072 cores clocked to 3 GB/s has been used. This paper will show a GPGPU implementation of the particular Kalman filter and the first results will be discussed.

1. INTRODUCTION

Usual system identification techniques for civil and mechanical structures assume the availability of large set of samples derived from a stationary quasi steady structure. On the opposite, several scenarios involve time varying structures. For example, due to interaction with aerodynamics in aeronautics, some critical parameter may have to be monitored, for unstability monitoring (leading possibly to flutter) of in flight data due to fuel consumption and speed change. This relates to the monitoring of time varying structural parameters such as frequencies and damping ratios.

The main difficulty in estimating the system parameters in a state space model is that in addition to the parameters, state variables are also unknown and unmeasured. Therefore it is important to use an approach that combines state and parameter estimation. It has been shown in [1] that extended Kalman filter based approach gives in general biased or divergent estimate. Moreover, in [2], a comparison between particle filter and EKF based on several examples shows that the particle filter gives a better parameter estimation, especially when non-linearity dealing with parameter estimation is significant. Thus, the particle filter is a good candidate for parameter estimation.This approach has been investigated for constant parameter estimation in [3] or non linear systems as in [4].

A typical particle filter algorithm consists in considering the state as a random variable following

a Markov process and approximating it by a cloud (set) of random particles mimicking its evolution according to the process equation. Each particle represents a possible state value, and at a given time it has a weight corresponding to its probability of being the most likely state value according to the measurement equation. At a given time, the estimated state is the weighted sum of all particles.

The particle filter yields to a good parameter estimation as shown in [2], but the drawbacks of considering the parameters as auxiliary state variable are first that the computation of the weight of a particle representing an augmented state is not accurate enough because of the nested dependencies between the state and the parameters, second that the number of particles needed to have an acceptable approximation explodes with the dimension of the state which makes the standard particle filter inefficient for high dimensional systems. A solution is to modify particle filter algorithm in order to take advantage of the fact that for linear systems, the state can be analytically marginalized out conditional on the parameter according to the system equations. Such filters have been introduced in [5, 6] as a Rao-Blackwellised particle filter and in [7] as an interacting Kalman filter. From the pure mathematical point of view, interacting Kalman filters can be encapsulated into particle filter models associated with Kalman predictor signals and virtual observation models. The main idea of an interacting Kalman filter is to consider particles evolving in the parameter space. For each particle, a corresponding state is estimated by applying a Kalman filter to the system in which parameter values are replaced by values associated to this particle. The weight of each particle is computed from the likelihood of the parameter sample it represents and its corresponding state. This results in a bank of adaptive Kalman filters combined with a particle filter. Such algorithm has been introduced and tested in [8]. In the present paper, GPGPU implementation of the particle evolution developed through the cloud2SM project [9] is investigated.

2. DYNAMICAL MODEL AND STRUCTURAL PARAMETERS

Consider a dynamical system of dimension n whose behavior is given by the following equation:

$$M_t\ddot{x}(t) + C_t\dot{x}(t) + K_t x(t) = \sigma u(t) \tag{1}$$

where M_t, C_t et K_t are respectively the time varying matrices of mass, damping and stiffness, $\sigma \in \mathbb{R}$ and u is the input force. u is modeled as a white Gaussian noise with time varying covariance matrix R^u. Dynamic of M_t, C_t et K_t is assumed to be unknown. Denote by

$$X(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix},\ A_t = \begin{bmatrix} 0 & \mathscr{I} \\ -M_t^{-1}C_t & -M_t^{-1}K_t \end{bmatrix} \text{ and } \Sigma_t = \begin{bmatrix} 0 \\ M_t^{-1}\sigma \end{bmatrix} \tag{2}$$

Equation (1) is equivalent to

$$\frac{\partial X(t)}{\partial t} = A_t X(t) + \Sigma_t u(t) \tag{3}$$

Denote by $X_k = X(k\delta)$, $M_k = M_{k\delta}$, $C_k = C_{k\delta}$, $K_k = K_{k\delta}$, $A_k = A_{k\delta}$, $u_k = u(k\delta)$ and $F_k = e^{\delta A_k}$. Suppose that for $t \in [k\delta,\ (k+1)\delta[$, $A_t = A_k$ and $\Sigma(t) = \Sigma_k$ and consider a time discretization at a rate δ. The discretized linear state-space model is

$$X_{k+1} = F_k X_k + \left(\int_0^\delta \exp(A_k v)\Sigma_k dv \right) u := F_k X_k + B_k u_k. \tag{4}$$

The measurement equation is

$$Y_k := D_k X_k + H_k v_k.$$

Now describe the structural characteristics of the system (1). As the poles of a time varying system are not defined, it is supposed that for each t the time-variation of the system are freezed in order to obtain for each t a different time-invariant system such that poles and the notion of damping can be considered. The use of these frozen poles has been introduced by [10]. As a consequence, it is the possible to

consider for each t the modes or eigenfrequencies μ and the associated eigenvectors Φ_μ of the system, which are solutions for $t_k = k\delta$ of

$$\begin{cases} \det[\mu^2 M_k + \mu C_k + K_k] &= 0, \\ [\mu^2 M_k + \mu C_k + K_k]\Phi_\mu &= 0. \end{cases} \tag{5}$$

Then associated the mode–shapes are obtained by $D\Phi_\mu$. The frequency and damping coefficients are

$$\mathbf{f} = \frac{b}{2\pi} \text{ (Hz)}, \ \mathbf{d} = \frac{|a|}{\sqrt{a^2+b^2}} \in [0,1] \text{ with } a = \Re(\mu) \text{ and } b = \Im(\mu) \tag{6}$$

For $i \in \{1 \dots n\}$, we denote by c_i, k_i and m_i respectively the time varying coefficients of damping, stiffness and mass, and we denote by

$$\theta_k = (c_1 \dots c_n, k_1 \dots k_n, m_1 \dots m_n) \text{ at } t_k = k\delta.$$

System (4) can be written as

$$\begin{cases} X_{k+1} &= F(\theta_k)X_k + B(\theta_k)u_k \\ Y_k &= D(\theta_k)X_k + H(\theta_k)v_k \end{cases} \tag{7}$$

This relation explicits the mapping between θ_k and the system matrices. At each k, $F(\theta_k)$ can be computed for θ_k using (2). Similar mapping are derived between θ_k and the other system matrices.

3. THE INTERACTING KALMAN FILTER

Interacting Kalman filter and more generally genetic type particle methods are a class of Monte Carlo methods to sample from complex high-dimensional probability distributions for estimating their normalizing constants. Thus, these methods can be used to recursively compute $\hat{\theta}_k$. In the context of parameter estimation, genetic type particle methods can be classified into two main algorithms, the particle filter algorithm [2] and the interacting Kalman filter algorithm [7] which is combination of a Kalman filter and a particle filter and is adapted for tracking the time varying parameters of a linear system.

Consider the state space systems (X_k, Y_k) and (θ_k, Y_k), X_k and θ_k follows a Markov process with respective transition law $P(X_k|X_{k-1})$ and $P(\theta_k|\theta_{k-1})$. The measurement process Y_k is such that

$$P(Y_{0:k}|X_{0:k}) = P(Y_{0:k}|X_k).$$

From a Bayesian point of vue an optimal Bayesian solution $\hat{X}_k$ (respectively $\hat{\theta}_k$) to the filtering problem of estimating X_k (respectively θ_k) is given by the mean of the conditional probability density $P(X_k|Y_{0:k})$ (respectively $P(\theta_k|Y_{0:k})$)

$$\hat{X}_k = E(X_k|Y_{0:k}) (\text{ respectively } \hat{\theta}_k = E(\theta_k|Y_{0:k})) \tag{8}$$

In the next section, a brief introduction to Kalman and particle filters algorithms is given. Then a description of how they are combined in the interacting Kalman filter algorithm is given.

3.1 The Kalman filter

Consider the linear state-space system (7), where the system matrices are assumed to be known for each k and assume that X_0 is a Gaussian random variable with a mean $\hat{X}_0$ and covariance matrix P_0. The optimal filter equations can be directly solved using the Kalman filter algorithm:

$$(\hat{X}_{k-1}, P_{k-1}) \xrightarrow{prediction} (\hat{X}_k^-, P_k^-) \xrightarrow{updating} (\hat{X}_k, P_k)$$

The prediction and updating steps are given by

$$\begin{array}{l} \hat{X}_k = \hat{X}_k^- + G_k(Y_k - D_k\hat{X}_k^-) \text{ and } P_k = P_k^- - G_k D_k P_k^- \\ \hat{X}_k^- = F_k\hat{X}_{k-1} \text{ and } P_k^- = F_k P_{k-1} F_k' + B_k R_k^w B_k' \end{array} \tag{9}$$

with the gain matrix $G_k = P_k^- D_k'(D_k P_k^- + H_k R_k^v H_k')^{-1}$.
In Equation 9, a transpose of a matrix M is denoted by M'.

3.2 The particle filter

In the case where the state-space system is non linear, namely if

$$X_{k+1} = f_k(X_k, u_k) \text{ and } Y_k = g_k(X_k, v_k)$$

with non linear functions f_k and g_k, optimal filter equations can not be solved explicitly, therefore explicit formulas for the derivation of $E(X_k|Y_{0:k})$ are not available. The solution is to compute a corresponding particle approximation. The main idea is to evolve a cloud of N independent random samples ξ_k^i termed particles with associated weights w_k^i such that

$$\sum_{i=1}^{N} w_k^i 1_{\xi_k^i} \to P(X_k|Y_{0:k}) \text{ as } N \to \infty$$

These particles follow an intuitive genetic mutation-selection algorithm as follows using a two step procedure called mutation/selection

3.2.1 The mutation

Let $\sum_{i=1}^N w_{k-1}^i 1_{\xi_{k-1}^i}$ be a particle approximation of the conditional probability density $P(X_{k-1}|Y_{0:k-1})$. Every particle ξ_{k-1}^i performs a local random move according to $P(X_k|X_{k-1})$, then a particle approximation of $P(X_{k-1}|Y_{0:k-1})$ is given by

$$\sum_{i=1}^{N} w_{k-1}^i 1_{\xi_{k^-}^i} \tag{10}$$

3.2.2 The selection

Evaluate the weights of the $\xi_{k^-}^i$ using:

$$w_k^i = \frac{w_{k-1}^i P(Y_k|X_k = \xi_{k^-}^i)}{\sum_{j=1}^N w_{k-1}^j P(Y_k|X_k = \xi_{k^-}^j)} \tag{11}$$

where $P(Y_k|X_k = \xi_{k^-}^i)$ is computed using the measurement equation.

When particle weights are concentrated on a few number of particles, namely when $N_{eff} = (\sum_{i=1}^N (w_k^i)^2)^{-1}$ is less than $0.9N$, all particles whose weight is less than $\frac{1}{N}$ are removed and the others are duplicated proportionally to their weights. After this resampling, all the weights are equal to $\frac{1}{N}$. The particle filter based algorithm can be described in the few steps of Algorithm 1.

Algorithm 1 Particle filter algorithm

0: [**Input:**]Initialization: Simulate N initial particles $\xi_0^i, i \in \{1,\dots N\}$ according to the probability density of X_0
1: **for** $k = 1$ to number of samples **do**
2: **for** $i = 1$ to number of particles **do**
3: Mutation (prediction): Every particle ξ_{k-1}^i performs a local random move according to $P(X_k|X_{k-1})$, then a particle approximation of $P(X_{k-1}|Y_{0:k-1})$ is given by (10)
4: Selection (update): Evaluate the weights of the $\xi_{k^-}^i$ using (11) and compute the conditional probability density of Y_k given X_k according to the measurement equation
5: **end for**
6: Compute N_{eff}
7: If N_{eff} is less than $0.9N$, remove non significant particles and resample
8: **end for**
8: [**Output:**] For each k, an approximation of $E(X_k|Y_{0:k})$ is $\hat{X}_k = \sum w_k^i \xi_k^i$

3.2.3 Remark

The particle filter algorithm is well adapted to state estimation [7]. In fact the process and measurement equations can solve directly the mutation and the selection steps. This is not trivially the case when it comes to parameter estimation. One solution (see [2]) is to rewrite the state-space system by including the parameters (stiffness and damping coefficients) as a part of the state vector. The consequence then is that the particles will have components corresponding to the state and components corresponding to the parameters, and the computation of their weight will not necessarily give a good indication on the likelihood of their parameter components. This is not the option considered in this paper. The approach combines Kalman filter to estimate the state X_k and particle filter estimating θ_k.

3.3 Interacting Kalman filter

In this section, a particle approximation of $\hat{\theta}_k = E(\theta_k|Y_{0:k})$ is derived. Notice that a particle approximation of $\hat{X}_k$ would involve a direct and simple relation between Y_k and X_k. For the particle approximation of $\hat{\theta}_k$, the corresponding relation is not as explicit as previously, and then the corresponding particles weights can not be computed as easily. The interacting Kalman filter provides an original method to evaluate these weights, combining both particles and a classical Kalman filter. In fact, notice that

$$P(Y_k|\theta_k) = \mathcal{N}(C_k(\theta_k)\hat{X}_k^{\theta,-}, C_k(\theta_k)P_k^{\theta,-} + R_k^v) \tag{12}$$

where $\hat{X}_k^{\theta,-}$ and $P_k^{\theta,-}$ are defined in (9) and also depend on θ.

The main idea of the algorithm is to associate to each particle π_i^k representing the parameter, a state estimate ξ_i^k associated to $X_k(\pi_i^k)$ computed recursively from a Kalman filter as described in Algorithm 2.

It is observed that measurements likelihoods are much more influenced by the stiffness coefficients than by their damping coefficients. This remark extends to the likelihood of the particles representing these quantities. It means that a perturbation on the stiffness coefficients causes an equivalent perturbation on the likelihood of particles while a perturbation on the damping coefficients causes a perturbation on a relatively much smaller scale.

Stiffness variation perturbs the estimation of damping, which is problematic considering that a good damping estimation is necessary especially in the context of instability. When the damping and stiffness components of a particle vary at the same time, the particle's weight (as computed in the interacting Kalman filter algorithm) will in practice almost only take into account the likelihood of its stiffness component and it is not obvious to distinguish among the particles, those whose damping component is most probable according to the observation likelihood. Estimating separately the stiffness, and reusing this stiffness estimate to estimate the damping, increase the sensitivity of our damping estimation procedure.

Algorithm 2 Interacting Kalman filter algorithm

0: [**Input:**]Initialization: Simulate N initial particles π_0^i and state estimates ξ_0^i, $i \in \{1,\dots N\}$ according to the probability densities of θ_0 and X_0
1: **for** $k = 1$ to number of samples **do**
2: **for** $i = 1$ to number of particles **do**
3: Mutation(1): $\pi_{k-1}^i \to \pi_{k^-}^i$, every π_{k-1}^i performs a local random move according to $P(\theta_k|\theta_{k-1})$
4: Mutation(2): $\xi_{k-1}^i \to \xi_{k^-}^i$: for each $\pi_{k^-}^i$, compute $F(\pi_{k^-}^i)$, $B(\pi_{k^-}^i)$, $D(\pi_{k^-}^i)$ and $H(\pi_{k^-}^i)$. Then using the Kalman filter equations (9) with these matrices, compute $\hat{X}_k^-(\pi_{k^-}^i)$ and $\xi_k^i = \hat{X}_k(\pi_{k^-}^i)$
5: Selection: Update the weights of $\pi_{k^-}^i$ as in (11) using (12)
6: **end for**
7: Compute N_{eff} and resample
8: **end for**
8: [**Output:**]For each k, an approximation of $E(\theta_k|Y_{0:k})$ is $\hat{\theta}_k = \sum w_i^k \pi_i^k$

The stiffness is firstly estimated and then frozen before computing the new damping estimate. In fact, if the particles have all their stiffness component at the same value (which has been already estimated), it is easier to sort the particles with respect to their damping component. The proposed algorithm is as follows: For each k, a particle system evolves to track the evolution of the stiffness component, then another particle system is evolved to track the damping component using the new stiffness component value. Then the algorithm is updated for another k.
Consider a decomposition of the parameters and the corresponding particles according to their stiffness/damping components

$$\pi_k^i = ((\pi_k^i)_C, (\pi_k^i)_K) \text{ and } \theta_k = ((\theta_k)_C, (\theta_k)_K)$$

In order to track efficiently the damping variation, decoupling the probability distributions of the damping/stiffness components of the particles is needed, such that, if $(\hat{\theta}_k)_K$ is an estimate of $(\theta_k)_K$, then $E((\theta_k)_C|Y_{0:k}, (\hat{\theta}_k)_K)$ is an estimate of $(\theta_k)_C$. The resulting algorithm uses in a coordinated manner two interacting Kalman filters in order to compute successive particle approximations of $P((\theta_k)_K|Y_{0:k}, (\hat{\theta}_{k-1})_C)$ then $P((\theta_k)_C|Y_{0:k}, (\hat{\theta}_k)_K)$.

Algorithm 3 Interacting Kalman filter algorithm with decoupled distributions

0: [**Input:**]Initialization: Simulate N initial particles π_0^i and state estimates ξ_0^i, $i \in \{1,\dots N\}$ according to the probability densities of θ_0 and X_0
1: **for** $k = 1$ to number of samples **do**
2: **for** $i = 1$ to number of particles **do**
3: Mutation(1): $(\pi_{k-1}^i)_K \to (\pi_{k^-}^i)_K$ according to $P((\theta_k)_K|\theta_{k-1})$
4: Mutation(2): Using the Kalman filter equations (9) where F, B, D and H are computed for $\theta = ((\pi_{k^-}^i)_K, (\hat{\theta}_{k-1})_C)$, $i \in \{1\dots N\}$
5: Selection: Update the weights $(w_k^i)_K$ using the relation (12) with $\theta = ((\pi_{k^-}^i)_K, (\hat{\theta}_{k-1})_C)$
6: **end for**
7: Resampling and approximation of $(\hat{\theta}_k)_K$
8: **for** $i = 1$ to number of particles **do**
9: Mutation(1'): $(\pi_{k-1}^i)_C \to (\pi_{k^-}^i)_C$ according to $P((\theta_k)_C|(\theta_{k-1})_C, (\hat{\theta}_k)_K)$
10: Mutation(2'): Using the Kalman filter equations (9) where F, B, D and H are computed for $\theta = ((\hat{\theta}_k)_K, (\pi_{k^-}^i)_C)$
11: Selection: Update the weights $(w_k^i)_C$ using the relation (12) with $\theta = ((\hat{\theta}_k)_K, (\pi_{k^-}^i)_C)$
12: **end for**
13: Resampling and approximation of $(\hat{\theta}_k)_C$
14: **end for**
14: [**Output:**] For each k, estimates of the structural parameters given by a particle approximation of

$$(\hat{\theta}_k)_K = E((\theta_k)_K|Y_{0:k}, (\hat{\theta}_{k-1})_C) \text{ then } (\hat{\theta}_k)_C = E((\theta_k)_C|Y_{0:k}, (\hat{\theta}_k)_K) \tag{13}$$

4. NUMERICAL RESULTS

Interacting Kalman filter as all Monte Carlo algorithms is suitable for parallel computation. In fact, the computations done for one particle are independent of all other particles, the algorithm was adapted in order to run on a parallel environment. A Matlab parallel toolbox on a 4 cpu (central processor unit) core machine was used showing already a performance and computational speed improvement was first obtained. The numerical application was firstly proposed in [2], then in [8], where it is borrowed from. A parallel GPU implementation is now investigated. Parameter estimation here will focus on coefficients of both stiffness and damping matrices. Frequencies and dampings can be deduced as in [8].

4.1 Four story shear building

For the numerical experiments the case study is the same as in [2], but the results show a significant improvement comparing to those obtained there. This case study is a four story shear building with equal masses $m_1 = m_2 = m_3 = m_4 = m$. The inter-story stiffness k_1, k_2, k_3 and k_4 and the inter-story viscous damping coefficients c_1, c_2, c_3 and c_4 are time varying. The mass matrix is $M = diag(m)$ and the stiffness and damping matrices are given by

$$C_t = \begin{bmatrix} c_{4t} & -c_{4t} & 0 & 0 \\ -c_{4t} & c_{4t}+c_{3t} & 0 & 0 \\ 0 & -c_{3t} & c_{3t}+c_{2t} & -c_{2t} \\ 0 & 0 & -c_{2t} & c_{2t}+c_{1t} \end{bmatrix} \text{ and } K_t = \begin{bmatrix} k_{4t} & -k_{4t} & 0 & 0 \\ -k_{4t} & k_{4t}+k_{3t} & 0 & 0 \\ 0 & -k_{3t} & k_{3t}+k_{2t} & -k_{2t} \\ 0 & 0 & -k_{2t} & k_{2t}+k_{1t} \end{bmatrix}$$

The evolution of the stiffness/damping coefficients is given by Figures 1 and 2 where the two top curves represent c_1 and c_2 (respectively k_1 and k_2) and the two lower curves represent c_3 and c_4 (respectively k_3 and k_4)

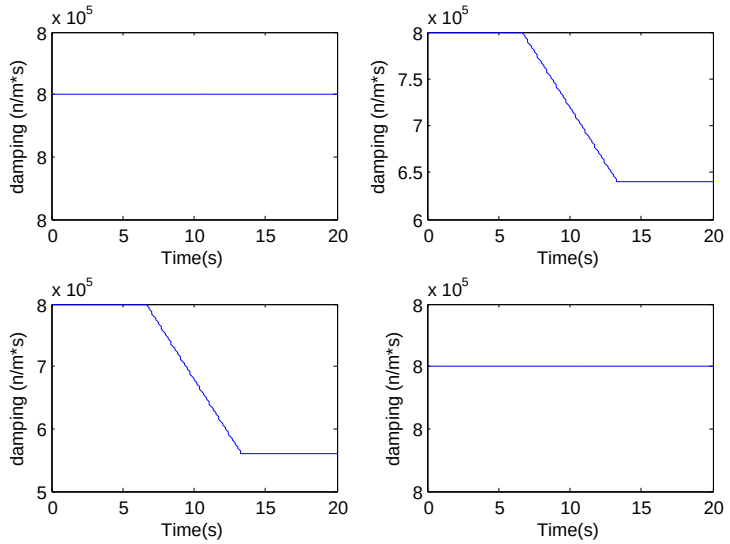

Figure 1: Damping coefficients

The excitation u used is a white noise of about 5% applied to the system. The observed output Y_k is the absolute acceleration time histories at the four stories. It is sampled at an interval of 0.02s. The values of observation Y_k used in the numerical experiments are simulated using Matlab. The parameters of the system are tracked between $t = 0$ and $t = 20s$ with 2000 particles. The rate of change of the damping and stiffness coefficient between two iterations is supposed not to exceed 2% of their value thus the variance v of the cloud of particles is such that $r = 2\%$.

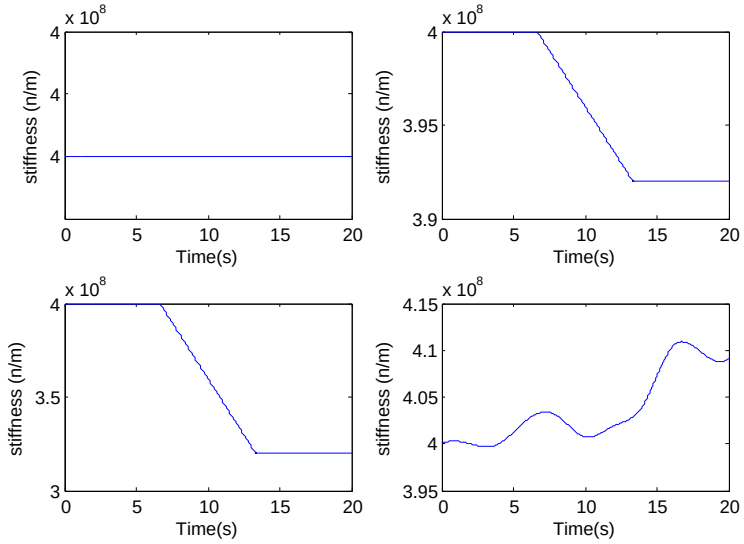

Figure 2: Stiffness coefficients

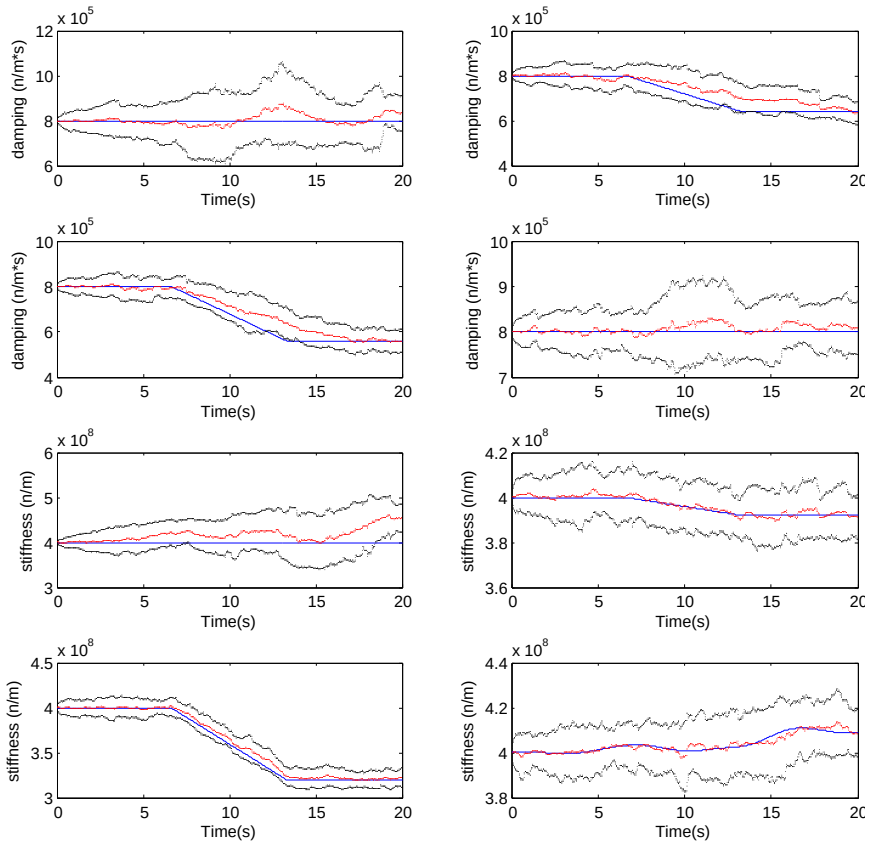

Figure 3: Exact (blue) and estimated (red) stiffness and damping coefficients

The top four curves of Figure 3 show the estimated evolution of the damping coefficients and the bottom four, the estimated evolution of the stiffness coefficients. The interval between the black curves represents two standard deviations on each side of the mean of the estimated parameters (weighted sum of the particles) corresponding to 95% of confidence level. This confidence interval shows the particle cloud

dispersion and its ability to track changes in the parameters. The size of the interval can be reduced at the expense of reactivity to changes.

4.2 First round of GPU optimisation

A GPU, or graphics processing unit is a processor type dedicated to massive parallel operation, first used for image processing, then after a while a certain amount of researchers and engineers have started to use their highly parallel behaviour to optimize general computing, this method is called GPGPU for General Purpose computing on GPU [11]. As a first implementation the main goal here is to execute concurrently particles operations from the Algorithm 3 in a GPU card, the new version of the algorithm designed to track parameters is presented in Algorithm 4.

Algorithm 4 Algorithm N^o 3 GPGPU version

0: [**Input:**]Initialization: Simulate N initial particles π_0^i and state estimates ξ_0^i, $i \in \{1,\dots N\}$ according to the probability densities of θ_0 and X_0
1: **for** $k = 1$ to number of samples **do**
2: KernelRandomize<<<NPARTICULES>>>(RandState)
3: KernelMuteStateAndParam<<<NPARTICLES>>>(RandState,Cloud,indices)
4: Resampling and approximation of $(\hat{\theta}_k)_K$
5: KernelRandomize<<<NPARTICLES>>>(RandState)
6: KernelMuteStateAndParam<<<NPARTICLES>>>(RandState,Cloud,indices)
7: Resampling and approximation of $(\hat{\theta}_k)_C$
8: **end for**
8: [**Output:**] For each k, estimates of the structural parameters given by a particle approximation of

$$(\hat{\theta}_k)_K = E((\theta_k)_K | Y_{0:k}, (\hat{\theta}_{k-1})_C) \text{ then } (\hat{\theta}_k)_C = E((\theta_k)_C | Y_{0:k}, (\hat{\theta}_k)_K) \tag{14}$$

A GPU function is called a kernel and launches a certain number of parallel threads, here equivalent to the number of particles contained in the cloud. For now two kernels have been used.

- **KernelRandomize** : It generates a random variable for each particle and parameter thanks to the cuRand library
- **KernelMuteStateAndParam** : It assesses the state and parameter mutation operations on each of the particles for different components of the parameter.
- **Resampling** : Those operations are still done on CPU and induce futile memory operation but they will be optimized in a second round.

For now we do not want to present here absolute results as further optimization of the algorithm is still under work, but we present here some comparative results about the algorithm execution time obtained thanks to a Quadro k6000 card of 3072 cores clocked to 3 GB/s, Table 1.

Table 1: Execution time for 2000 particles

Method	Matlab	C++	C++ openMp	MultiThread SSE	GPU
Time	1h	20min	5min	2min	70s

Thanks to this first round of optimization we succeed to improve slightly the execution time of the algorithm. For GPU optimization, it is important to group threads as power of two, it is an hardware recommendation. To compute a GPU kernel one has to define a grid which will contain blocks of threads. For example the considered card accepts a maximum size of blocks of 1024 threads. By adjusting both grid and block size the impact the number of particles used is decreased, see Figure 4. Contrary to the

GPU, for more than 2000 particles the CPU implementation, in Table 1, will have computation time which increases linearly. Those different results show that the GPU optimization can be a useful tool for particle filtering for SHM.

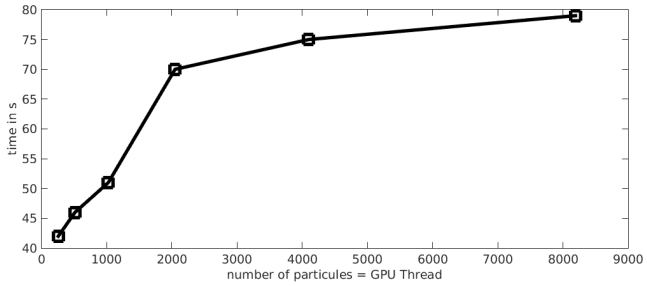

Figure 4: Time evolution against number of particles for custom kernel implementation

5. CONCLUSION

The problem of parameter estimation for linear time varying systems is addressed. The implementation of the algorithm described in [8] has been investigated. The efficiency of the interacting Kalman filter with decoupled probability distribution is shown through numerical simulations. However, further improvements could still be made to the algorithm, especially in terms of speed and genericity. A first implementation has been optimized using GPU. As it cannot give yet absolute results we have investigated the execution time relatively to the former algorithm and the results are promising. The next step will be to improve the memory management as well as the algorithm itself to obtain a faster implementation and increase the number of degrees of freedom.

REFERENCES

[1] L. Ljung. Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems. *IEEE Transactions on Automatic Control*, 24:36–50, 1979.

[2] J. Ching, J. L. Beck, and K. A. Porter. Bayesian state and parameter estimation of uncertain dynamical systems. *Probabilistic Engineering Mechanics*, 21:81–96, 2006.

[3] E. N Chatzi and A. W. Smyth. Particle filter scheme with mutation for the estimation of time-invariant parameters in structural health monitoring applications. *Structural Control and Health Monitoring*, 20(7):1081–1095, 2013.

[4] E. N Chatzi and A. W. Smyth. The unscented kalman filter and particle filter methods for nonlinear structural system identification with non-collocated heterogeneous sensing. *Structural control and health monitoring*, 16(1):99–123, 2009.

[5] A. Doucet, N. de Freitas, K. P. Murphy, and S. J. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. *UAI '00 Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*.

[6] P. Li, R. Goodall, and V. Kadirkamanathan. Estimation of parameters in a linear state space model using a rao-blackwellised particle filter. *IEE Proc., Control Theory Appl*, 151:727738, 2004.

[7] P. Del Moral. *Feynman-Kac formulae. Genealogical and interacting particle approximations*. Probability and Applications. Springer New York, 2004.

[8] M. Zghal, L. Mevel, and P. Del Moral. Modal parameter estimation using interacting kalman filter. *Mechanical Systems and Signal Processing*, 49(3), May 2012.

[9] A . Crinière, J. Dumoulin, L. Mevel, G. Andrade-Barosso, and M. Simonin. The cloud2sm project. In *EGU General Assembly*, 2015.

[10] L. A. Zadeh. Frequency analysis of variable networks. *Proceedings of the Institute of Ratio Engineers*, 38:291 299, 1950.

[11] N.Witt. *The CUDA Handbook A Comprehensive Guide to GPU Programming*. 2013.